UNIVERSITÉ DE SHERBROOKE

Projet Robus Armus

Rapport de conception

Rapport présenté à :

l'équipe professorale de S1

Par :

L'équipe P11, Gotham Project :

Adrien Burghgraeve #08358074

Gan Laporte #07242316

Jérémie Faucher-Goulet #08341911

Mathieu Foucault #07206739

Mathieu Fournier #08381235

Nicolas Roy #08365501

Pascal Guay #07197446

Steve Pelchat #08346881

Université de Sherbrooke

3 décembre 2008

Table des matières

Page Titre i

Table des matières ii

Introduction 1

Énoncé du problème 1

Présentation des options de solution 2

Formulation de la solution 3

Planification 5

Description des tâches 6

Échéancier 8

Liste des priorités 9

Portrait final 9

Conception matérielle 9

Conception logicielle 15

Tests 23

Conclusion 27

Références 28

Annexe 1 : Manuel de service 1

Annexe 2 : Guide de l'utilisateur 1

Annexe 3 : Code Source du robot 1

Introduction

Considérant la baisse des inscriptions en génie année après année, l'équipe professorale de S1 donne comme mandat à ses étudiants d'élaborer un robot qui intéresserait les enfants à la science. Évidemment, nous profitons de l'occasion pour donner un but éducatif à celui-ci, puisqu'un robot peut-être amusant et éducatif à la fois. Toutefois, la conception d'un projet de cette envergure comporte plusieurs étapes. Il y a d'abord la génération d'une solution à la problématique, puis vient ensuite la planification, la conception matérielle, la conception logicielle et les tests effectués. En annexe, vous trouverez le guide de l'utilisateur et le manuel de service.

Énoncé du problème

La tâche soumise en début de section consiste en quelques points bien précis. Tout d'abord, il faut construire un robot autonome capable d'interagir avec des enfants en bas âge, répondant aux critères du concours Robot-Jouet. Par autonomie, on considère un robot capable de se comporter de manière sécuritaire sans aucun contrôle extérieur. D'ailleurs, pour être autonome, il doit posséder sa propre alimentation énergétique. Par interaction, il est question d'un robot capable d'interagir avec les enfants. Selon les critères du projet, le robot doit interagir en faisant appel à un minimum de 2 sens. Les critères à respecter pour le robot sont :

- Respecter le thème de l'exposition permanente « Au fil des saisons » du musée de la nature et des sciences de Sherbrooke. Dans cette exposition, il y est présenté la flore et la faune du sud du Québec en choisissant une des sections correspondant à l'une des quatre saisons.
- Se déplacer de façon autonome dans un milieu courant, peu importe sa configuration.
- Se comporter de manière courtoise.
- Susciter l'intérêt de l'enfant.
- Interagir de façon compréhensible du point de vue de l'enfant.
- Être robuste et sécuritaire.
- Permettre d'évaluer la progression de l'enfant.
- Être documenté de façon adéquate (mode d'emploi, manuel de service).
- Être facile d'utilisation.

- Doit parler ou émettre des sons.
- Doit être éducatif.
- Être capable d'interagir d'au moins 2 façons différentes avec l'humain.
- Minimum de 4 réactions différentes à une bonne ou mauvaise réponse.
- Minimum de 4 questions et maximum de 10.
- Le robot doit choisir de façon aléatoire ses questions parmi son répertoire.
- Esthétique simple, mais attirante.
- Des points supplémentaires seront accordés pour la qualité du déplacement du robot.

Les règlements du concours à respecter sont les suivantes :

- Les équipes participantes doivent avoir un maximum de 10 membres.
- Il est possible d'obtenir de l'aide extérieure pour l'esthétisme et la mécanique du robot.
- Les dimensions maximales du robot sont de 1.5 x 1.5 x 2.5 pieds, cette dernière mesure représentant la hauteur. Un poids maximal de 45 lb est imposé.
- Les commandites sont autorisées et gérées par les équipes. Toutefois, ces démarches ne sont pas en lien avec le concours robot-jouet. De plus, aucun espace spécial n'est offert à cette fin.
- Une fiche descriptive du robot est requise.
- Un espace de 8 pieds par 6 pieds est disponible pour l'installation d'un kiosque de présentation par équipe.
- Il est possible que des médias demandent une démonstration.
- L'utilisation de matériel bruyant ou salissant, ainsi que la distribution de nourriture ou d'objet sont interdites.

En respectant ces règlements, critères et exigences, nous aurons répondu au problème.

Présentation des options de solution

Pour la création d'un robot-jouet, il est possible de fabriquer un animal ou une plante de la faune et la flore du sud du Québec. L'animal et la plante présentent plusieurs aspects qui peuvent répondent à la problématique. Le plus important est de choisir un robot capable d'attirer l'attention d'un enfant. Cet aspect est important à considérer. Puisque le robot doit être attrayant et capter l'attention d'un enfant, il faut choisir quelque chose qui attirera son attention. Le sens que nous croyons le plus

pratique est la vue, puisqu'il agit sur de longues distances. Un autre sens important chez un enfant est l'ouïe. Il s'agit d'une bonne façon d'interagir avec l'enfant en lui donnant des instructions et des informations dans un cadre éducatif et ludique. La variété de sons que l'enfant peut percevoir permet une multitude d'effets sur les capacités intellectuelles de l'enfant. Les interactions sonores sont pratiquement infinies. Le toucher est le troisième sens qui facilite l'apprentissage chez un enfant. Ce sens peut être utilisé tant sur une plante que sur un animal. Les sens du goûter et de l'odorat ne sont pas beaucoup associés à la mémoire, ce qui les rend beaucoup moins attrayants. Bien sur, il est possible d'obtenir une certaine forme de divertissement avec le goûter ou l'odorat. Les cinq sens en association avec un robot sont les solutions trouvées pour choisir et construire un robot-jouet respectant les spécifications de la problématique ainsi que les règlements.

Formulation de la solution

En fonction des solutions trouvées précédemment, nous avons dû les comparer pour nous permettre de choisir un animal ou une plante qui correspondait le mieux et qui s'adaptait le mieux aux solutions retenues. Le sens du toucher, de la vue et de l'ouïe sont les meilleures solutions pour les conditions initiales de la problématique. L'odorat est un sens difficile à stimuler avec un robot et cela n'attire pas vraiment l'attention des enfants. Le sens du goût est une bonne idée, mais les règlements nous interdisent de donner de la nourriture aux enfants. Cela aurait été une bonne idée de récompenser les bonnes réponses par des bonbons. Les trois autres solutions sont excellentes dans un cadre ludique et éducatif. Le mieux est d'utiliser ces trois solutions en même temps pour permettre un apprentissage et divertissement plus efficace chez l'enfant. En fonction de ces solutions retenues, nous avons choisi un animal au lieu d'une plante. Pour nous, un animal est plus attirant pour un enfant et se déplace plus naturellement qu'une plante. C'est visuellement plus attirant qu'une plante qui bouge peu. Après avoir choisi de concevoir un animal, il est ressorti quatre idées principales d'animaux. Les

avantages et les inconvénients de la grenouille, de la chauve-souris, de la luciole et de la tortue sont présentés dans le tableau comparatif suivant.

Grenouille		Chauve-souris	
Pour	Contre	Pour	Contre
Gonflement de la gorge	Pas assez de mouvements	Ultra-sons buts : éducatif et technique	Laid
Simple	Non attrayant	Ailes qui bougent	Complexité
Coassement	Moins original	Plus original	
Luciole		Tortue	
Pour	Contre	Pour	Contre
DEL pour le derrière de la luciole	Besoin d'agrandir	Simple (forme de demi-sphère)	Encombrant
Lumières attirent les enfants	Ces ultra-sons fragiles	Solide et sécuritaire	Réaction lente
	Ressemble à une mouche (répugnant)		Association rapide avec ennui

La chauve-souris nous a semblé le meilleur choix en fonction des arguments cités plus haut. En effet, cela nous a semblé être une idée originale. Les ultra-sons ont joué un grand rôle dans la décision de l'animal, parce que cela fait correspondre la réalité mécanique à la nature. En effet, la chauve-souris et le robot fonctionnent sur le même principe des ultra-sons par l'utilisation des sonars. On tire partie du déplacement à l'aide des sonars pour permettre à un enfant de faire bouger le robot seulement en passant sa main à proximité. Cela stimule donc la vue de l'enfant et permet une interaction physique avec le robot. Le battement des ailes a également joué un très grand rôle dans le choix de la chauve-souris. Il est possible de faire battre les ailes d'une luciole, mais le

battement d'ailes d'une chauve-souris est plus imposant, donc plus attrayant visuellement pour un enfant. Pour répondre à la problématique, il faut aussi s'assurer que la chauve-souris respecte les règlements et les critères initiaux. Plusieurs de ces critères sont remplis dans les fonctions du programme qui permet à la chauve-souris d'augmenter ses interactions.

On veut donc développer une chauve-souris qui peut battre des ailes et éviter les obstacles. Celle-ci serait d'une taille suffisamment grande pour pouvoir bien interagir avec les enfants et pas trop grande pour pouvoir se déplacer sans encombrement. Le corps du robot devra être assez solide pour résister à tout assaut de la part d'un enfant. La chauve-souris devra aussi poser des questions aux enfants qui concernent la chauve-souris à l'état sauvage.

Pour conclure cette section, les idées retenues sont nombreuses et chacune a des avantages et des inconvénients, mais nous trouvions plus facile d'intégrer les sens d'interaction choisis à la chauve-souris. Il va être plus facile pour nous de remplir les critères du concours et d'optimiser les interactions avec l'enfant.

Planification

La section suivante du rapport contient la planification, donc un échéancier ainsi que les tâches et les personnes associées à ces tâches. L'importance et la priorité de chacune de ces tâches sont aussi discutées dans cette section. Le tableau suivant énumère les responsables des différentes tâches entourant la conception du robot jouet. Il s'agit de tâches administratives et de conception. Une description sommaire suit le tableau et décrit les tâches en question.

Responsable logistique	Mathieu Fournier
Secrétaire	Gan, Adrien

Modérateur	Gan, Steve
Mécanisme de battement des ailes	Mathieu Foucault, Jérémie, Adrien
Structure des ailes	Mathieu Fournier, Nicolas
Revêtement et coupe du tissu	Nicolas, Mathieu Fournier
Conception de la coquille	Adrien, Steve
Magasinier	Mathieu Foucault
Responsable de la programmation	Pascal
Circuits électroniques	Nicolas, Jérémie
Bonne entente	TOUS

Les responsables des tâches suivantes ont la responsabilité de s'assurer de l'exécution du travail dans les délais prévus. Les autres membres de l'équipe ont l'obligation d'aider et de participer le plus possible aux différentes tâches à exécuter.

Description des tâches

Responsable logistique

Doit s'assurer du transport, de l'entreposage et de la commande des pièces. Il doit aussi trouver les locaux pour le travail en équipe. Son travail s'effectue tout au long du projet.

Secrétaire

C'est la personne responsable de prendre des notes de l'avancement du robot jouet. C'est aussi celui qui s'assure de la remise des travaux.

Modérateur

Il doit s'assurer du respect de l'échéancier. Il doit aussi s'assurer que le travail avance et qu'il est bien fait. Il s'occupe d'empêcher les dérapages lors des travaux en équipe.

Mécanisme de battement des ailes

Il doit présenter et mettre au point un mécanisme permettant de faire battre les ailes.

Structure des ailes

Doit trouver une façon de fabriquer des ailes qui représente bien celle de la chauve-souris.

Revêtement et coupe du tissu

Doit s'assurer des bonnes dimensions du tissu, des types de tissus utilisés et de la fixation du tissu au robot.

Conception de la coquille

Doit concevoir une coquille qui recouvre le robot et qui est en mesure d'accueillir les ailes ainsi que la fourrure du robot.

Magasinier

Responsable de se procurer les matériaux nécessaires à la conception du robot. C'est aussi la personne qui est responsable de l'achat de ces matériaux.

Responsable de la programmation

Avec l'aide de l'équipe, il doit trouver les idées réalisables pour la programmation du robot. Il doit aussi guider et subdiviser les tâches de programmations. C'est aussi lui qui fixe les délais associés à la programmation.

Circuits électroniques

Il est responsable de la conception des circuits et d'assister les soudures à effectuer. Il effectue un contrôle de qualité sur les circuits soudés. Il est aussi responsable de trouver les façons les plus efficaces pour parvenir à nos fins.

Échéancier

Voici l'échéancier qui a été utilisé pour la réalisation du projet.

Activités	13 oct.	20 oct.	27 oct.	3 nov.	10 nov.	17 nov.	24 nov.	4 déc.
Squelette de la chauve-souris	X	X						
Finition apparence chauve-souris		X	X	X				
Fabrication mécanisme pour bouger les ailes			X	X				
Programmer mouvements de base	X	X						
Programmer mouvements des ailes				X	X			
Programmer sonars ultrasoniques		X	X					
Écrire les messages et questions dictés par le robot	X	X	X					
Enregistrer les messages et questions				X				
Programmer les interactions vocales avec boutons					X	X		
Tests, déverminage et finition du robot				X	X	X	X	
Rédaction du rapport de conception, du guide de service et d'utilisateur							X	X
Conception et montage du kiosque							X	X

Liste des priorités

Voici la liste des tâches énoncées dans l'échéancier. Celles-ci sont basées sur le principe que le robot doit être fonctionnel pour la journée de présentation du 4 décembre. C'est pourquoi les fonctionnalités de base sont prioritaires.

1. Programmation des mouvements de bases.
2. Structure de la chauve-souris et mécanisme des ailes.
3. Programmer les sonars et le mouvement des ailes.
4. Écrire les questions et les programmer.
5. Conception du kiosque.
6. Rédaction des rapports.

Portrait final

Notre robot est assez fidèle au modèle fixé au départ. En effet, il bat des ailes tout en se déplaçant, sauf que nous avons ajouté différents types de déplacements. Il se déplace en évitant les objets comme nous l'avions planifié, mais il se déplace aussi en suivant les objets qu'il détecte. Il pose 9 questions de façon aléatoire aux enfants et enregistre les réponses dans un fichier afin de compiler des statistiques. De plus, nous avons réalisé un jeu de Simon à l'aide des boutons de couleurs situés sur le robot.

Conception matérielle

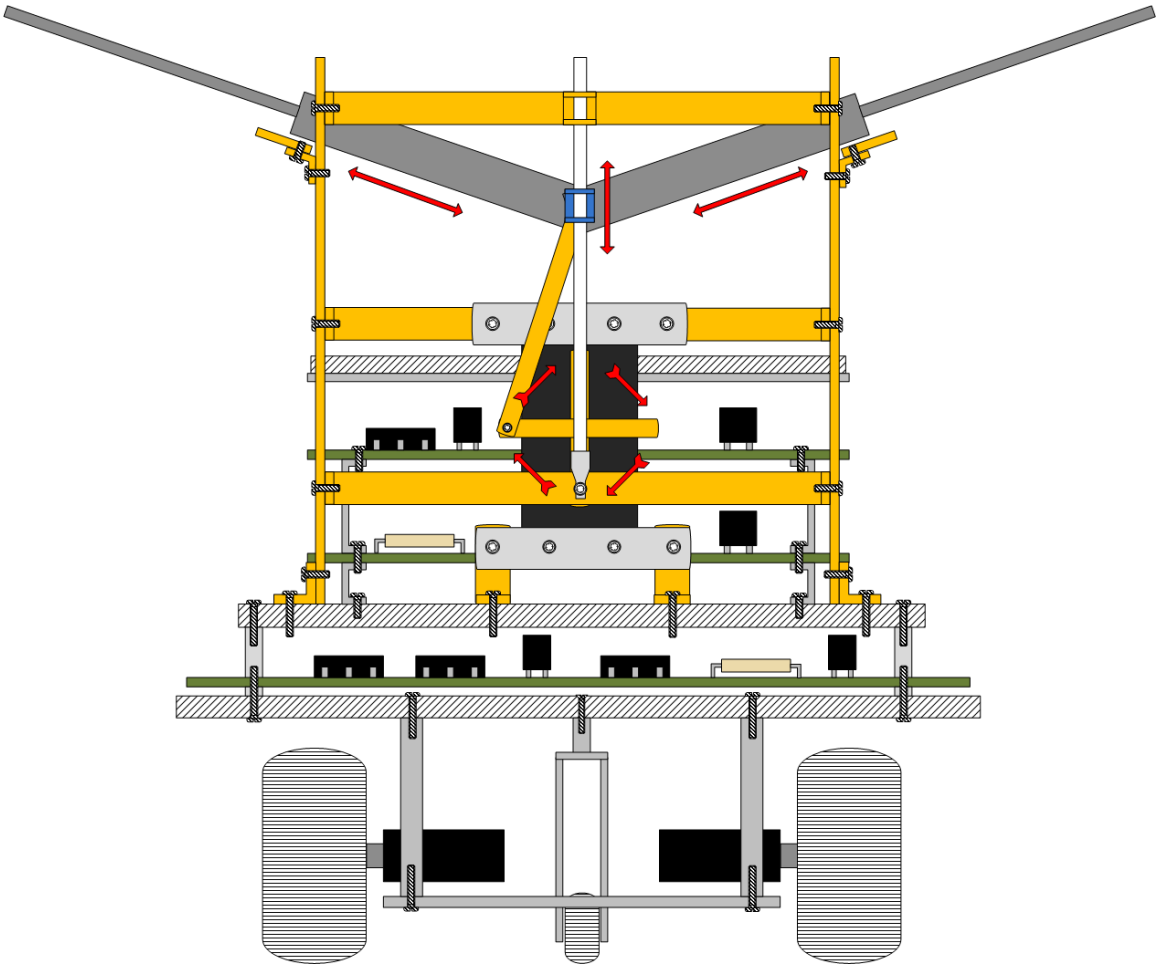
Le choix de la plate-forme Robus-Armus s'est imposé de lui-même. De plus, cette plate-forme est particulièrement adaptée pour le présent concours, avec plusieurs capteurs et plusieurs fonctions déjà programmées pour l'utilisation de dispositifs et de capteurs mis à notre disposition.

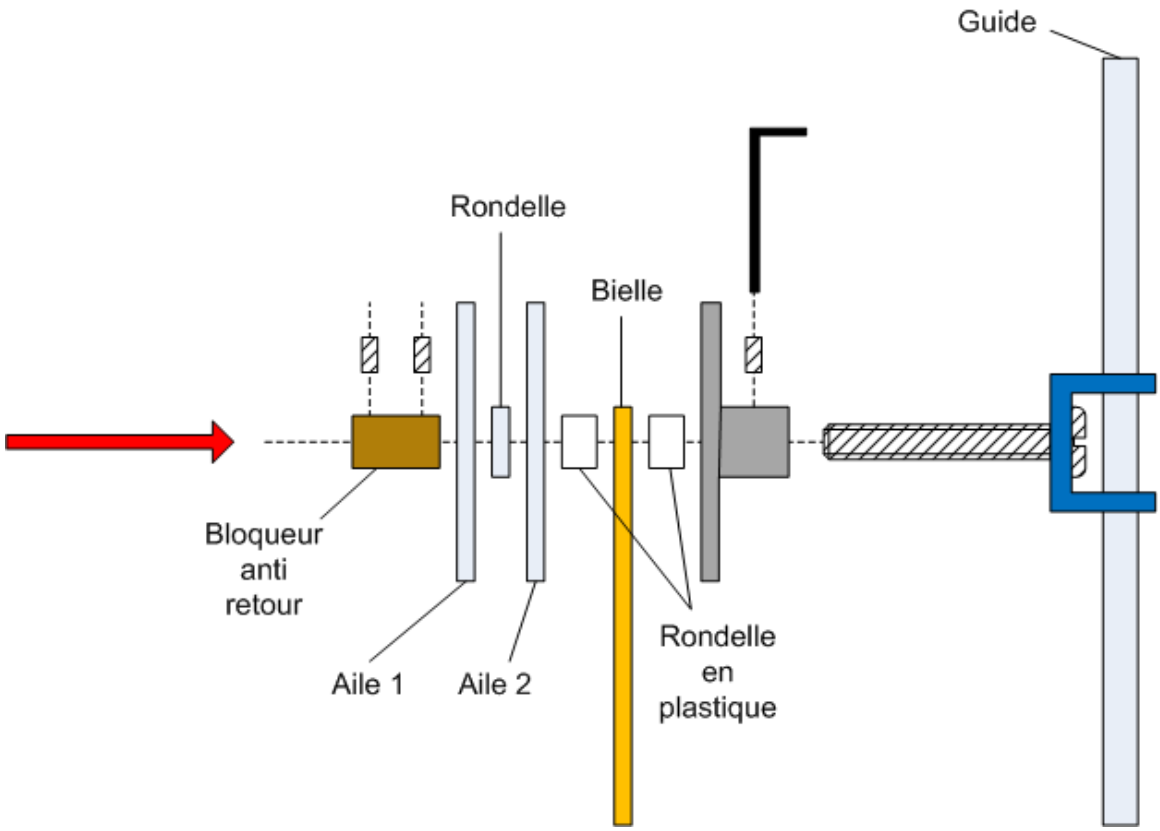
Pour l'assemblage de la plate-forme Robus, la documentation fournie a rendu le processus relativement facile à comprendre. L'assemblage de Robus nécessitait toutefois beaucoup de soudure pour les personnes à leur début en soudure. L'une des plaques fut complètement fonctionnelle dès le départ, alors que l'autre nécessita beaucoup de déverminage. De nombreux tests furent effectués afin de vérifier le montage, avant d'y connecter l'ordinateur et les autres cartes pour les premiers tests de l'assemblage électronique de la plate-forme. Une fois la plate-forme assemblée, la partie créative du robot commençait.

Avec le choix d'une chauve-souris comme robot, il fallait d'abord s'interroger sur la forme et la taille de celle-ci. Sachant que nous voulions faire des ailes, nous avons tout de suite choisi de faire le corps le plus petit possible, pour obtenir des ailes pas trop longues et qui gardent une bonne proportionnalité avec le corps. Ensuite, nous avons choisi le matériau dans lequel nous allions fabriquer cette coque. Nous avons opté pour le papier mâché appliqué sur de la cage à poule comme moule pour sa facilité de construction et de manipulation. En même temps, il fallait installer un support pour le système de battement d'ailes sur la plate-forme mécanique du robot. Afin d'économiser l'espace au maximum et de faire notre robot le plus petit possible, nous avons choisi de reconfigurer le montage proposé d'empilement de cartes électroniques. On a apposé la plaque de plexiglas déjà de la bonne taille directement au dessus de Robus pour s'en servir comme support.

Nous nous sommes servis de cette plaque afin de fixer notre support interne. Ainsi, nous étions le plus bas possible. Des trous furent percés dans cette plaquette de plexiglas afin d'y visser une structure dessus, faite à partir de Meccano. Les Meccanos furent choisis pour leur facilité d'utilisation (aucune scie ni soudure requise) et leur solidité. Beaucoup de montage et de démontage furent nécessaires avant d'obtenir un support qui permettait aux ailes de bien se mouvoir. Par la suite, un support vissé au plexiglas au dessus d'Armus fut installé pour le moteur. Il ne restait alors qu'à raccorder le moteur aux ailes, ce qui fut une tâche beaucoup plus difficile que prévu. En effet, les

recherches furent longues pour trouver la bonne combinaison de boulons et de vis afin d'éviter que les ailes se dévissent lors de leur fonctionnement. Il fallait également garder un léger espace afin de permettre une rotation libre au niveau de la bielle qui aurait été impossible si le tout avait été bien serré. Le système d'ailes/moteur fonctionnait donc sur la base suivante :



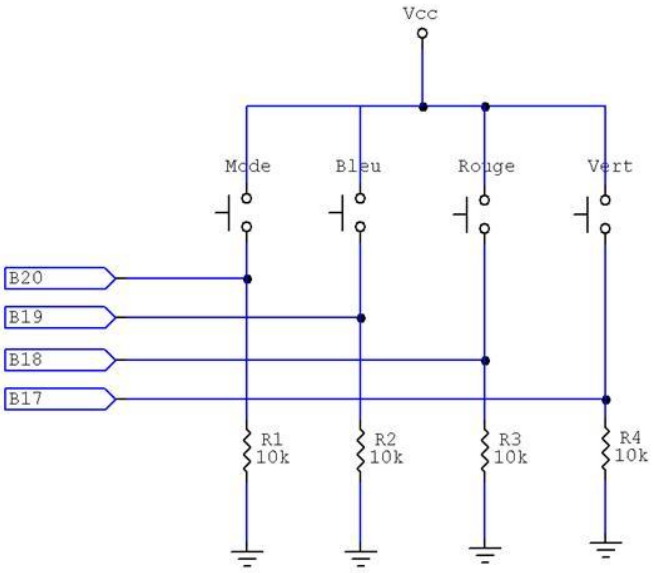


Par la suite, une autre tâche difficile apparut lorsque vint le temps d'installer l'électronique sous la coque. L'espace limité par notre choix de faire la taille du robot le plus petit possible a rendu la tâche assez difficile. Plusieurs essais furent tentés, jusqu'à ce qu'une solution permette un maximum d'optimisation de l'espace à l'intérieur de la coque afin de laisser un espace accessible sur le dessus pour le câblage. Après l'accomplissement de cette tâche, la structure mécanique et informatique était complétée, ce qui inclut évidemment la coque extérieure.

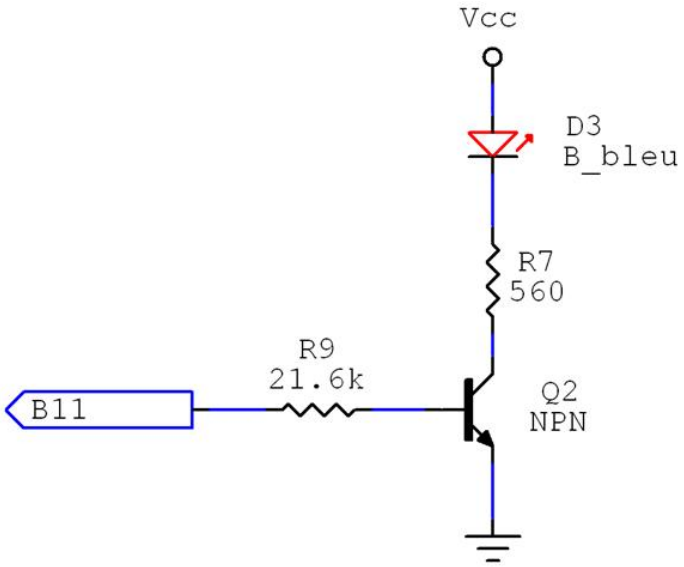
C'était maintenant l'heure d'habiller l'extérieur de la coque de notre robot, afin de lui donner l'apparence souhaitée. Pour se faire, une protubérance au niveau du museau fut modelée par l'ajout de matière bourrant (sac de plastique, boule de papier, etc.) maintenu en place par un large ruban collant gris. Un tissu noir fut collé sur la coque sur la superficie du visage, puis le reste du corps fut recouvert d'un tissu-fourrure trouvé en magasin. Les trous pour l'ajout des boutons et des lumières électroluminescentes furent percés, ainsi qu'une trappe au dessus du robot pour donner accès à l'intérieur. La flexibilité et la tension de la coque suffisent à maintenir

celle-ci en place sur la plateforme de manière suffisamment serrée pour qu'elle ne bouge pas. Les boutons et les lumières installés, il ne restait plus qu'à bâtir les circuits et câbler l'intérieur du robot afin de contrôler tous les périphériques depuis l'intérieur. Les circuits furent assemblés sur le « breadboard ». Un connecteur DB-25 fut utilisé afin de faire les liens entre tous les fils qui passent de la coque au châssis pour permettre le démontage de la coque. Une fois toutes ces étapes accomplies et corrigées à maintes reprises en cours de route, nous étions prêts à passer à la programmation.

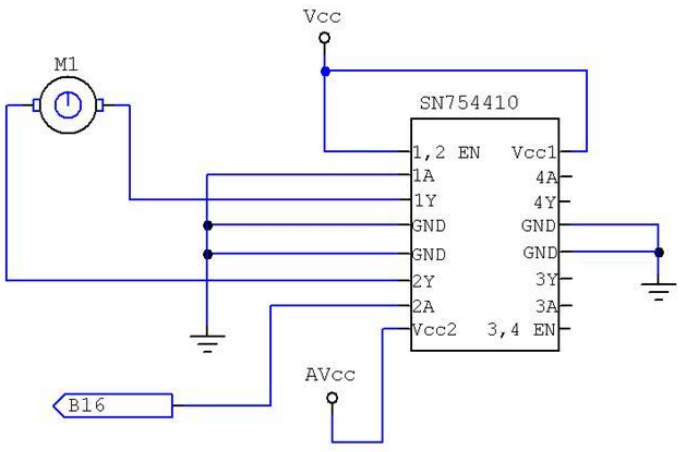
Circuit des boutons :



Circuit des DELs :

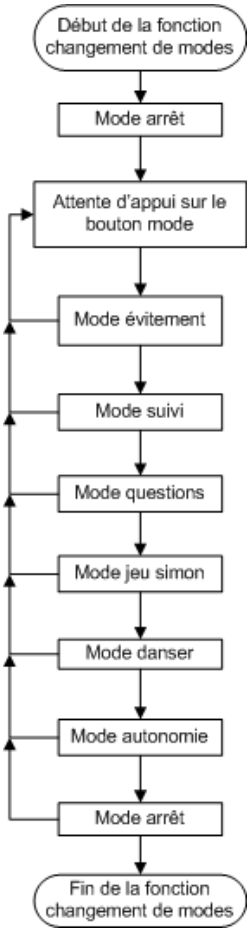


Circuit du Pont H :



Conception logicielle

La conception logicielle du robot repose sur 7 modes différents. Ces modes sont destinés à rendre le robot autonome et à éduquer l'enfant en lui permettant de s'amuser. Les modes sont changés comme suit:

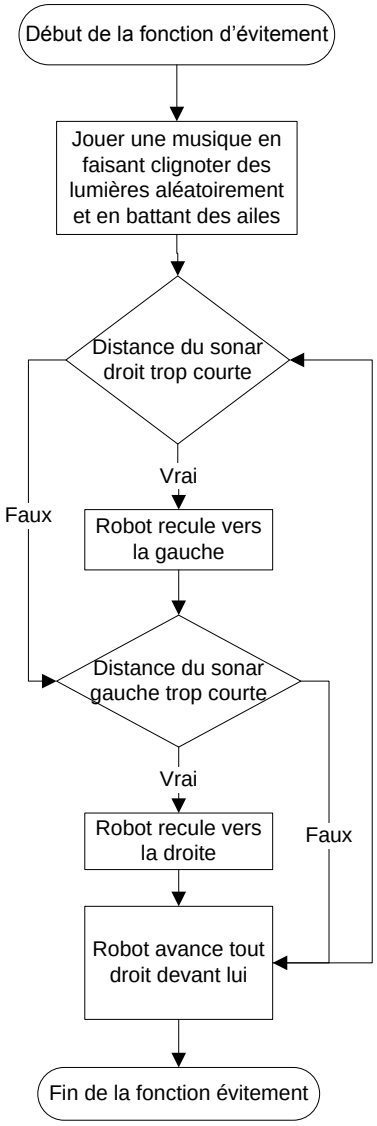


Voici l'explication de la conception logicielle de chacun de ces modes.

Mode arrêt

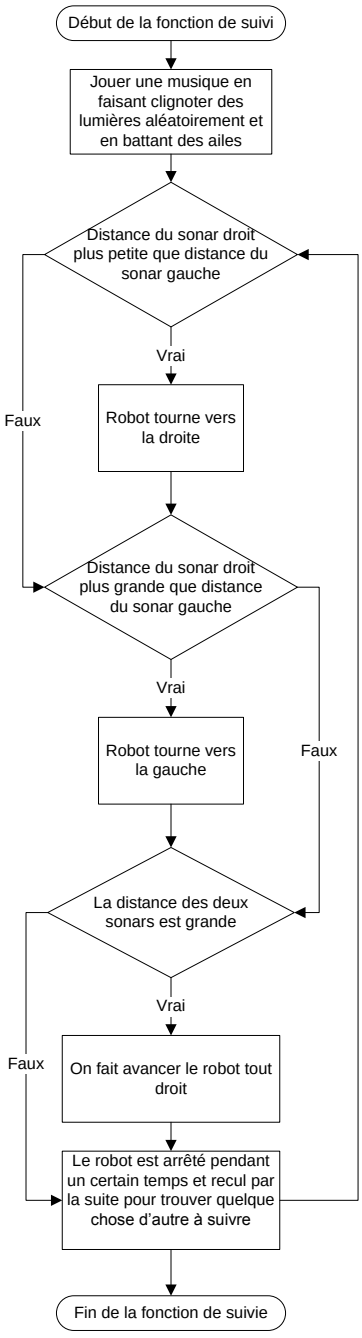
Dans ce mode, le robot est arrêté, il n'y a que ses yeux qui sont allumés. Il ne possède aucune interaction et attend que le bouton mode soit pressé afin de passer au mode suivant.

Mode évitement


Début de la fonction d'évitement
Jouer une musique en faisant clignoter des lumières aléatoirement et en battant des ailes
Distance du sonar droit trop courte
Vrai
Faux
Robot recule vers la gauche
Distance du sonar gauche trop courte
Vrai
Faux
Robot recule vers la droite
Robot avance tout droit devant lui
Fin de la fonction évitement

Tout d'abord, le robot commence par battre des ailes tout en faisant jouer de la musique et en faisant clignoter aléatoirement ses lumières. Ces actions ont pour but de le rendre plus vivant aux yeux des gens. Le robot vérifie ensuite la distance mesurée par le sonar droit et le sonar gauche. Si cette distance est trop courte, le robot tourne tout en reculant afin d'éviter l'obstacle auquel il fait face. Lorsque la distance mesurée par les sonars est suffisamment grande, le robot avance tout droit.

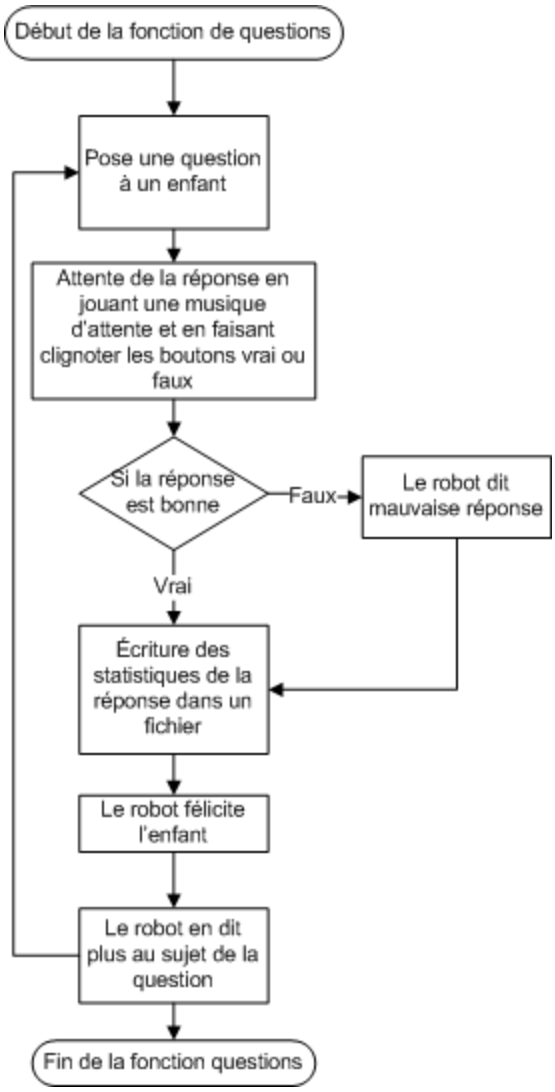
Mode suivie


Début de la fonction de suivi
Jouer une musique en faisant clignoter des lumières aléatoirement et en battant des ailes
Distance du sonar droit plus petite que distance du sonar gauche
Vrai
Faux
Robot tourne vers la droite
Distance du sonar droit plus grande que distance du sonar gauche
Vrai
Faux
Robot tourne vers la gauche
La distance des deux sonars est grande
Vrai
Faux
On fait avancer le robot tout droit
Le robot est arrêté pendant un certain temps et recul par la suite pour trouver quelque chose d'autre à suivre
Fin de la fonction de suivie

Comme c'est le cas pour la fonction évitement, le robot fait jouer une musique différente tout en battant des ailes et en faisant clignoter aléatoirement ses lumières afin de lui donner l'impression qu'il est vivant. Dans ce mode, le robot vérifie la distance

du sonar droit et la compare avec celle du sonar gauche. Si la différence de la distance entre les deux sonars est plus grande que 20 centimètres, alors le robot tourne en direction du sonar qui mesurait la distance la plus courte. Lorsque le robot s'arrête devant un obstacle, il vérifie si celui-ci est immobile pour évaluer s'il fait face à un mur par exemple. Le robot s'arrête pendant un certain temps et si la distance de l'objet ne change pas, alors il recule en tournant pour lui permettre de suivre une autre cible. Finalement, si une distance assez grande est mesurée par les sonars, il avance tout droit jusqu'à ce qu'il trouve une nouvelle cible à suivre.

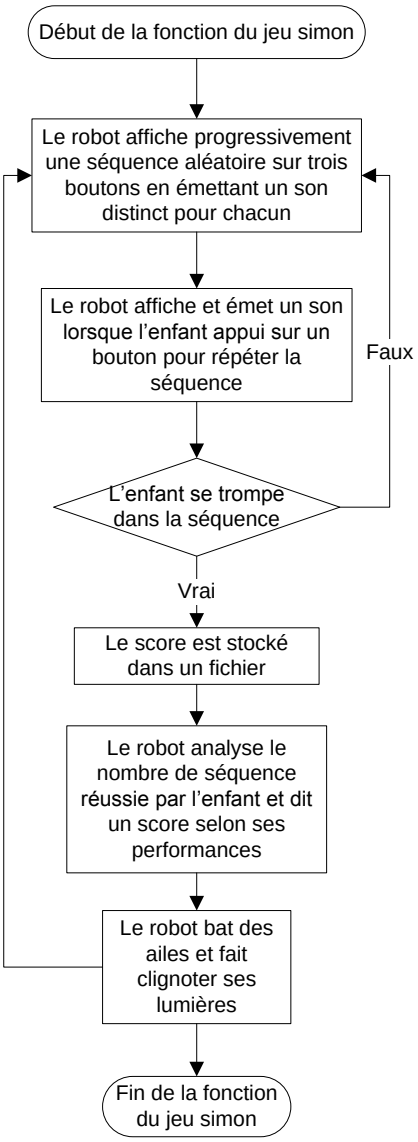
Mode questions


Début de la fonction de questions
Pose une question à un enfant
Attente de la réponse en jouant une musique d'attente et en faisant clignoter les boutons vrai ou faux
Si la réponse est bonne
Faux
Le robot dit mauvaise réponse
Vrai
Écriture des statistiques de la réponse dans un fichier
Le robot félicite l'enfant
Le robot en dit plus au sujet de la question
Fin de la fonction questions

Ce mode sert à éduquer l'enfant à propos des chauves-souris. Le robot pose 9 questions à l'enfant dans un ordre aléatoire sans répéter aucune question. Afin de permettre cela, il est nécessaire de stocker les questions déjà posées par le robot dans un tableau afin qu'elles ne se répètent pas. Le robot attend ensuite la réponse de la personne avec une musique d'attente, en faisant clignoter le bouton vert et rouge qui correspondent respectivement à vrai ou faux. Lorsque la personne appuie sur un bouton, la réponse de l'enfant est enregistrée dans un fichier de compilation des résultats appelés reponses.txt. Le robot vérifie si la réponse à cette question est bonne

et félicite l'enfant tout en battant des ailes et en faisant clignoter ses lumières. Si la réponse n'est pas bonne, le robot dit à l'enfant que c'est la mauvaise réponse. Le robot dit ensuite plus d'informations à propos de la question à l'enfant.

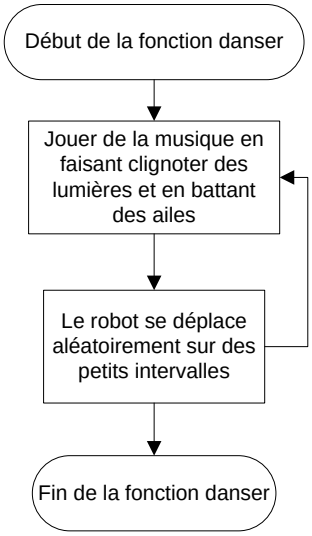
Mode jeu Simon


Début de la fonction du jeu simon
Le robot affiche progressivement une séquence aléatoire sur trois boutons en émettant un son distinct pour chacun
Le robot affiche et émet un son lorsque l'enfant appui sur un bouton pour répéter la séquence
Faux
L'enfant se trompe dans la séquence
Vrai
Le score est stocké dans un fichier
Le robot analyse le nombre de séquence réussie par l'enfant et dit un score selon ses performances
Le robot bat des ailes et fait clignoter ses lumières
Fin de la fonction du jeu simon

Le jeu Simon est un jeu de répétition de séquences qui permet d'amuser l'enfant tout en permettant d'exercer sa mémoire. Le robot affiche progressivement une

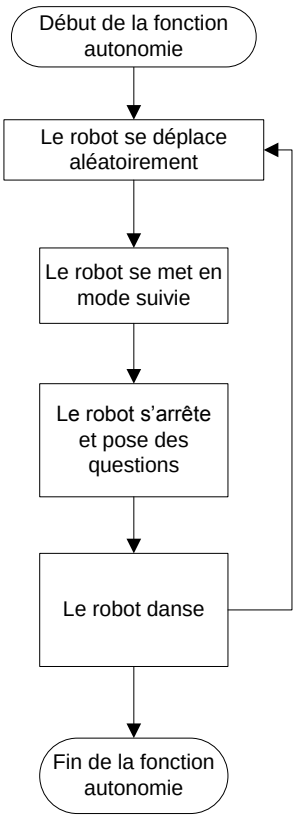
séquence aléatoire sur ses boutons rouge, bleu et vert en émettant un son pour chacun. L'enfant doit ensuite répéter la même séquence que le robot vient de lui montrer. La séquence devient de plus en plus longue au fur et à mesure que l'enfant réussit à les répéter. Lorsque l'enfant se trompe, le robot enregistre le score obtenu par l'enfant dans un fichier simon.txt, analyse le nombre de séquences réussies et lui dit un commentaire sur ses performances. Si l'enfant a réussi une série de séquences inférieure à 10, le robot dit « Bravo, mais tu peux t'améliorer », tandis que pour un résultat supérieur à 10, il dit « bravo tu es bon». Le robot bat ensuite des ailes et fait clignoter les lumières pour féliciter l'enfant.

Mode danse


Début de la fonction danser
Jouer de la musique en faisant clignoter des lumières et en battant des ailes
Le robot se déplace aléatoirement sur des petits intervalles
Fin de la fonction danser

Ce mode fait danser le robot sur une musique qui bouge. Le robot bat des ailes et fait clignoter les lumières tout en se déplaçant aléatoirement sur de petites distances afin de lui donner l'impression de danser.

Mode autonomie


Début de la fonction autonomie
Le robot se déplace aléatoirement
Le robot se met en mode suivie
Le robot s'arrête et pose des questions
Le robot danse
Fin de la fonction autonomie

Dans le mode autonomie, le robot se déplace aléatoirement dans la pièce pendant un certain temps pour ensuite tomber en mode suivi. Le robot suit une personne et s'arrête pour lui poser trois questions. Lorsque la personne a fini de répondre aux trois questions, le robot se met à danser pendant 20 secondes pour ensuite continuer le cycle. Ce mode permet au robot de passer lui même par différentes actions afin de le rendre autonome pour qu'il puisse interagir avec les gens.

Il a fallu ajuster plusieurs fonctions dans notre programme afin qu'il puisse se conformer aux limitations des performances de la plate-forme robotique Armus/Robus. En effet, il a fallu réduire le nombre d'actions simultanées que le robot pouvait faire ou bien réduire sa vitesse d'exécution afin de ne pas surcharger son bus de transfert de

données. Cette surcharge du bus de données nous donnait l'erreur SPI_INIT et faisait arrêter certaines fonctionnalités du robot aléatoirement lors de ses performances. Il est donc important de limiter le plus possible les possibilités d'une telle erreur.

Tests

Lors de la conception du robot, nous avons eu à accomplir certaines tâches afin de parvenir au but ultime qui est la réalisation de notre chauve-souris. Pour être certains que chacune des tâches a bien été réalisée, nous avons effectué plusieurs tests afin de nous assurer du bon fonctionnement du robot. Les tests ont été faits par rapport aux plaquettes Robus et Armus, le corps de la chauve-souris, la structure des ailes, le mécanisme de battement des ailes, l'électronique ainsi que la programmation finale de la chauve-souris.

Nous avons commencé par nous assurer du bon fonctionnement de la plaquette Robus. Pour ce faire, nous avons suivi les procédures dans le manuel d'assemblage et d'utilisation de Robus et d'Armus. Nous nous sommes d'abord assuré que les tensions d'alimentation étaient bien celles indiquées dans le guide. Ensuite, nous avons testé chacun des moteurs dans chacun de leur sens. Une fois les moteurs des roues fonctionnels, nous avons alors pu tester les encodeurs de roue avec l'oscilloscope. Une fois les ajustements effectués, le syntoniseur de fréquence (PLL), l'oscillateur ainsi que les capteurs infrarouges ont été testés. Ensuite, nous avons testé l'amplificateur audio en y injectant un signal de 1 KHz. Ces tests se sont très bien déroulés puisque nous n'avions aucune conception à effectuer pour cette partie.

Les premiers tests de programmation ont été effectués par la suite. Ceux-ci ont été effectués avant la mise à jour du logiciel d'Armus. Il y avait donc de nombreux

problèmes lors de la programmation. Une fois les mises à jour effectuées, tout était fonctionnel.

Pour ce qui est du corps de la chauve-souris, nous avons conclu, lors d'une réunion, de la faire en papier mâché dû à son poids minime et à sa facilité de fabrication. Pour le faire, nous avons dû effectuer un premier test sur la cage à poule (notre soutien), car nous ne connaissions pas la difficulté de la tâche. La première n'ayant pas la bonne forme, une deuxième a été conçue. Nous avions maintenant notre corps de la chauve-souris. Nous savions cependant qu'en utilisant une structure en papier mâché, nous devions faire un certain compromis entre la robustesse et la solidité. Notre robot serait probablement plus fragile que les autres.

Pour la structure des ailes, nous avons fait plusieurs tentatives afin d'arriver à notre produit final. Notre but était d'avoir les ailes les plus solides possible, tout en gardant une bonne rigidité. Tout d'abord, nous avons utilisé des cintres en métal afin d'obtenir la structure des ailes voulue. Considérant leur poids, nous avons estimé que nous devions utiliser des tiges en aluminium pour obtenir plus de légèreté. Une fois les tiges en aluminium trouvées, nous avions à les réunir ensemble afin d'obtenir la structure des ailes finale. Nous voulions tout d'abord souder toutes ces tiges afin d'avoir des ailes solides. Cependant, des techniciens en génie mécanique nous ont dit que cela n'était pas possible. Finalement, nous avons utilisé un principe de serres afin de garder les tiges en place. La longueur des ailes faisait en sorte de faire forcer davantage le moteur. En utilisant une structure en aluminium, nous avons gagné au niveau du poids. Nous savions cependant qu'elles ne devraient pas battre très vite pour ne pas les endommager, car elles restaient fragiles. La structure des ailes était alors terminée. Il ne nous suffisait alors qu'à les recouvrir d'un tissu pour obtenir l'esthétisme voulu.

Les ailes étant terminées, nous devions nous attarder au mécanisme de battement des ailes. Nous savions que cette partie serait probablement la plus difficile du robot. Pour nous faciliter la tâche, nous avons décidé d'utiliser un système de meccanos pour sa facilité d'utilisation et pour un montage fiable et solide que cela nous donnerait. Nous avons effectué plusieurs combinaisons de meccanos afin de faire fonctionner le mécanisme. Notre but était d'éliminer le plus de friction possible, car notre principale limitation venait de la force du moteur. Nous étions également limités à la tension d'utilisation du moteur puisque qu'il y avait un seuil de 5,5 V à dépasser afin d'amorcer la rotation du moteur. Évidemment, notre première tentative n'était pas très efficace puisqu'il y avait beaucoup de friction. Nous avons cependant gardé le même principe de fonctionnement tout au long de nos tentatives, mais toujours en l'améliorant. Mesurant le courant de consommation du moteur et l'estimant à environ 250 mA, nous avions finalement réussi à obtenir un système fiable à long terme et qui consomme le moins de courant possible.

Pour ce qui est de l'électronique, cela comporte plusieurs parties, dont un pont en H, des transistors pour le DELs, un petit circuit pour les boutons poussoirs ainsi que les sonars. Nous avons choisi un pont en H pour sa facilité d'utilisation et parce qu'il y en avait déjà un sur la plaquette Robus pour les roues. Nous avions donc le circuit intégré à notre disposition. Il a très bien fonctionné du premier coup. Ensuite, le circuit pour les diodes électroluminescentes a très bien fonctionné également. Cependant, nous avons dû tester différentes valeurs de résistances afin de trouver un niveau d'intensité lumineuse adéquat pour nos DELs, car nous ne voulions pas éblouir les enfants lors de la présentation. Un petit circuit résistif a été mis en place pour faire fonctionner nos boutons poussoirs, rien de bien compliqué. Finalement, nous avons testé nos sonars, un élément essentiel au bon fonctionnement de notre robot. Les sonars utilisés nous étaient fournis par le département. Nous n'avions donc pas le choix des sonars. La position des sonars devait être la plus haute possible pour ne pas interférer avec le sol.

Nous n'étions pas limités avec les distances d'opération du sonar puisque sa plage de distances était amplement suffisante (moins de 5 cm à plus de 3 mètres). Ils ont très bien fonctionné et leur simplicité d'utilisation nous a favorisé la tâche (après avoir effectué les mises à jour de la plaquette Armus). Pour permettre une grande facilitation de branchement et pour notre espace restreint, nous sommes venus à la conclusion que nous devions souder tous les composants électroniques sur un connecteur DB-25.

Une fois la structure de notre chauve-souris terminée, il ne nous restait qu'à effectuer la programmation. Nous avons constaté qu'il y avait souvent des défaillances, ce qui nécessitait fréquemment des tests (tests de déplacement simple, tests de son, tests de sonars, tests des entrées / sorties de la plate-forme Armus). Tous ces tests ont été effectués indépendamment afin de s'assurer qu'aucun autre élément ne venait interférer. Une fois tous les éléments fonctionnels, nous avons élaboré un programme pour les différentes fonctionnalités du robot. Nous voulions mettre beaucoup d'emphase sur le côté interaction avec l'enfant. C'est pour cette raison que nous avons décidé d'insérer un jeu ainsi que différents modes (chasse, aléatoire, questions). Le mode chasse permettait à l'enfant d'avoir un rôle à jouer pour les déplacements du robot. Le mode aléatoire faisait danser notre chauve-souris, ce qui, nous croyions, augmenterait l'intérêt de l'enfant pour notre robot. Nous avons donc effectué les premiers tests avec le bloc d'alimentation fourni afin de ne pas occasionner d'autres problèmes. Une fois les déplacements fonctionnels ainsi que tous les autres éléments du robot, il était alors le temps de tout tester, mais de façon autonome, c'est-à-dire avec les batteries. Ce changement s'est très bien déroulé, malgré nos craintes par rapport à la consommation du moteur des ailes. Finalement, un test global a été effectué en milieu similaire à la présentation pour s'assurer du bon fonctionnement du robot et de voir les possibles défaillances qui pourraient survenir. Quelques correctifs furent apportés, mais rien de majeur.

Après la réalisation de la programmation, notre robot était alors 100% fonctionnel et nous étions donc près à la compétition. Nous avions donc toutes les chances de notre côté.

Conclusion

En fin de compte, l'échéancier fut assez bien respecté, si ce n'est que certains retards qui ont été compensés par l'avancement plus rapide de d'autres sections. De plus, il est à noter que la programmation de ce matériel fut très ardue puisqu'il fallait optimiser le code pour éviter les erreurs du type SPI_Init() qui indiquent une surcharge dans le bus de données. Le système de battement des ailes a nécessité beaucoup plus de temps qu'initialement prévu, considérant les nombreux ajustements nécessaires afin d'assurer un bon glissement et un mécanisme qui fonctionne à la perfection. Avoir eu plus de temps, nous aurions aimé ajouter des détecteurs supplémentaires sur les autres côtés du robot afin de permettre une meilleure détection des obstacles. Sommes toutes, nous sommes assez satisfaits du résultat.

Références

Références électroniques

Pour le Pont H

http://focus.ti.com/lit/ds/symlink/sn754410.pdf

Pour le transistor 2N3904

http://www.fairchildsemi.com/ds/MM/MMBT3904.pdf

Références manuscrites

Guide étudiant – S1

Manuel d'assemblage Robus

Annexe 1 : Manuel de service

Préface

Le projet présenté dans ce manuel est un robot jouet conçu par les étudiants au baccalauréat en génie électrique en première session. Ce robot est en fait une chauve-souris qui a pour but d'intéresser les enfants en bas âge à l'ingénierie. Notre chauve-souris présente donc plusieurs modes afin de réaliser cet objectif (ces modes sont présentés dans le guide de l'utilisateur).

Dans ce manuel, vous retrouverez le principe de fonctionnement du robot, les différents ajustements à effectuer, son principe de montage et de démontage, un procédé de réparation ainsi que plusieurs schémas qui pourraient vous être utiles.

Principe de fonctionnement :

Électrique :

*tous les schémas électriques sont fournis en annexe

Contrôle du moteur des ailes :

Le moteur des ailes est contrôlé par un pont en H via une sortie de la plaquette Armus. En fait, il sert à donner suffisamment de courant au moteur pour fonctionner. Ce n'est donc pas la sortie de la plaque Armus qui fournit le courant, mais plutôt l'alimentation.

Circuit des DELs :

Les DELs sont insérées dans un petit circuit à transistor. Ce dernier fonctionne en commutation, c'est-à-dire qu'il ne possède que deux modes : saturé ou bloquant. Le courant dans les diodes est contrôlé par une résistance en série avec ces diodes. Bien entendu, la résistance à la base du transistor sert à faire saturer le transistor.

Circuit des boutons :

Dans le circuit des boutons, il y a une résistance (reliée à la référence) en parallèle avec chaque bouton. Cela évite que les entrées de la plaquette Armus ne soient flottantes lorsqu'un bouton n'est pas enfoncé, car la référence est reportée à l'entrée. Lorsque le bouton est enfoncé, il y a 5V à l'entrée et un faible courant dans la résistance dû à sa grande valeur.

Mécanique :

Mécanisme des ailes

Les ailes peuvent battre à l'aide d'un moteur en rotation. En fait, l'axe de rotation du moteur est allongé afin d'augmenter l'amplitude des ailes. Cet axe est relié

aux ailes, ce qui les fait battre. On obtient ainsi un mouvement de haut en bas des ailes, ainsi qu'un mouvement de gauche à droite. Pour éliminer ce mouvement de gauche à droite, on utilise un guide qui limite le déplacement des ailes de haut en bas. Pour soutenir les ailes, on utilise également un support de chaque côté afin de les garder à une certaine hauteur.

Le connecteur

Tous les circuits électriques qui se doivent d'être utilisés aux entrées / sorties de la plaquette Armus sont reliés avec un connecteur DB-25 afin de faciliter le démontage de la coque.

Plaquette d'expérimentation (breadboard)

Toutes les connexions des circuits électriques (à l'exception des sonars) sont reliées via la plaquette d'expérimentation afin de faciliter les possibles modifications des circuits électriques. Elle est disponible par une trappe située sur le dos de la chauve-souris.

Disposition des plaquettes

La disposition des plaquettes est conçue afin de maximiser l'espace. L'ordre des plaquettes en commençant par le bas est : Robus, Armus, la carte I / o et la plaque d'expérimentation.

Logiciel :

La programmation du robot se fait à l'aide du logiciel C++. Il est possible de le programmer directement avec un ordinateur ou à l'aide d'une clé USB. Lorsque le robot démarre, les ailes ainsi que toutes les DELs sont activent (démarrage d'Armus). Ensuite, il faut peser sur le bouton user situé sur la plaque d'expérimentation afin d'intégrer le programme au robot. Ensuite, il est prêt à l'utilisation.

Ajustements :

Il est possible d'ajuster le volume des haut-parleurs à l'aide de potentiomètre R55 de la plaquette Robus.

Il est possible d'ajuster la tension d'alimentation du moteur des ailes (AVcc) à l'aide du potentiomètre R16. Il est très fortement déconseillé de descendre cette tension en bas de 6V, car le moteur des ailes aura de la difficulté à tourner convenablement.

Principe de montage

Insérer la coque en faisant attention de ne pas accrocher les plaquettes électroniques. La coque doit être positionnée afin que les ailes puissent être appuyées à l'endroit prévu.

Insérer les ailes en utilisant dans l'ordre : espaceur, première aile, rondelle, deuxième aile, bloqueur. Le tout doit être suffisamment serré, tout en évitant les frottements.

Branchez le connecteur DB-25 reliant les circuits électroniques. Il faut également s'assurer que les connexions soient adéquates sur la plaquette d'expérimentation, ainsi que les connexions sur la carte I/O.

Principe de démontage

- Débranchez le connecteur DB-25
- Enlevez les ailes
- Enlevez la coque en prenant soin de ne pas accrocher les plaques électroniques

Procédures de réparation

Si les DELs ne fonctionnent pas, vous pouvez vérifier les connexions sur la plaquette d'expérimentation, telles que vues en annexe. Vous pouvez également regarder si le signal se rend correctement à la carte I / o. Finalement, vérifiez que le connecteur DB-25 est bien branché.

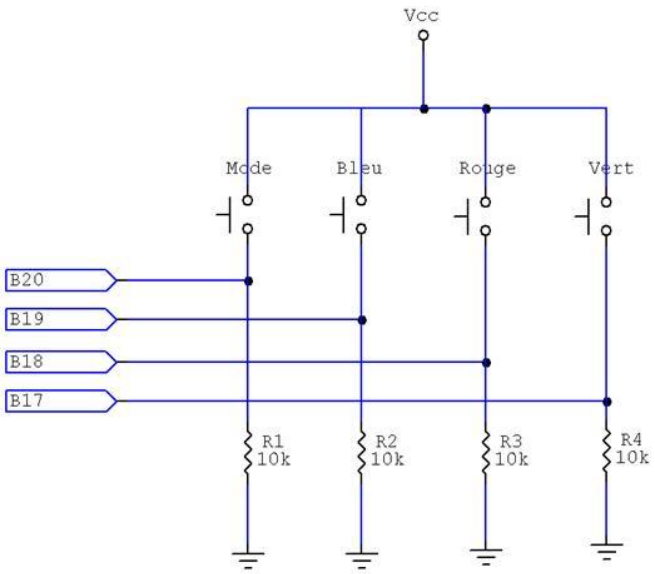
Si les boutons ne fonctionnent pas, vous pouvez appliquer la même procédure de vérification que pour les DELs.

Si les sonars ne fonctionnent pas, vérifiez que leurs connexions sont bien faites et qu'ils sont connectés dans le bon sens.

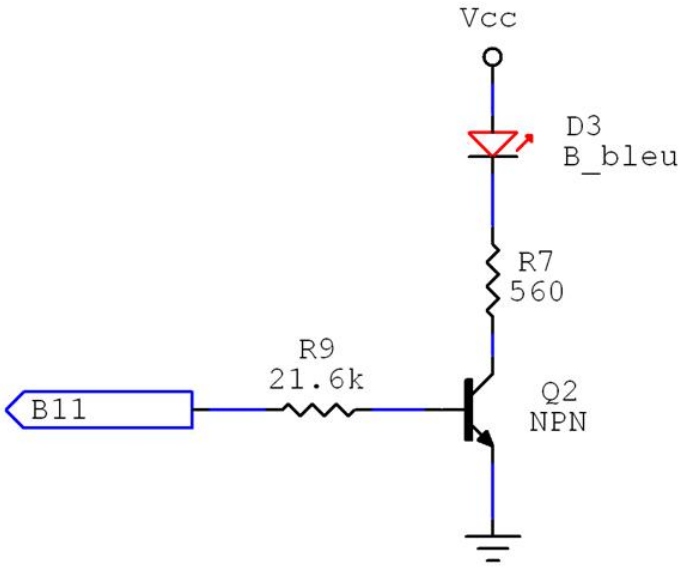
Si le moteur des ailes ne fonctionne pas, vérifiez d'abord que le signal logique du pont en H contrôlant le moteur est bien relié à la plaque I / o. Ensuite, vérifiez le circuit électrique du pont en H.

Annexes :

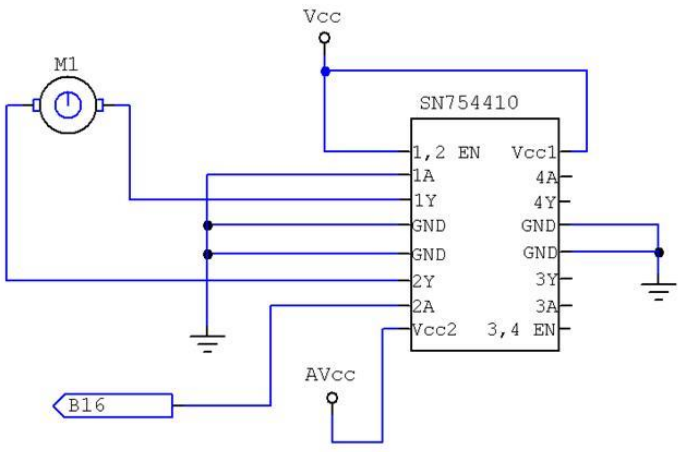
Circuit des boutons :



Circuit des DELs :



Circuit du Pont H :



Annexe 2 : Guide de l'utilisateur

Guide de l'utilisateur

Démarrage du robot :

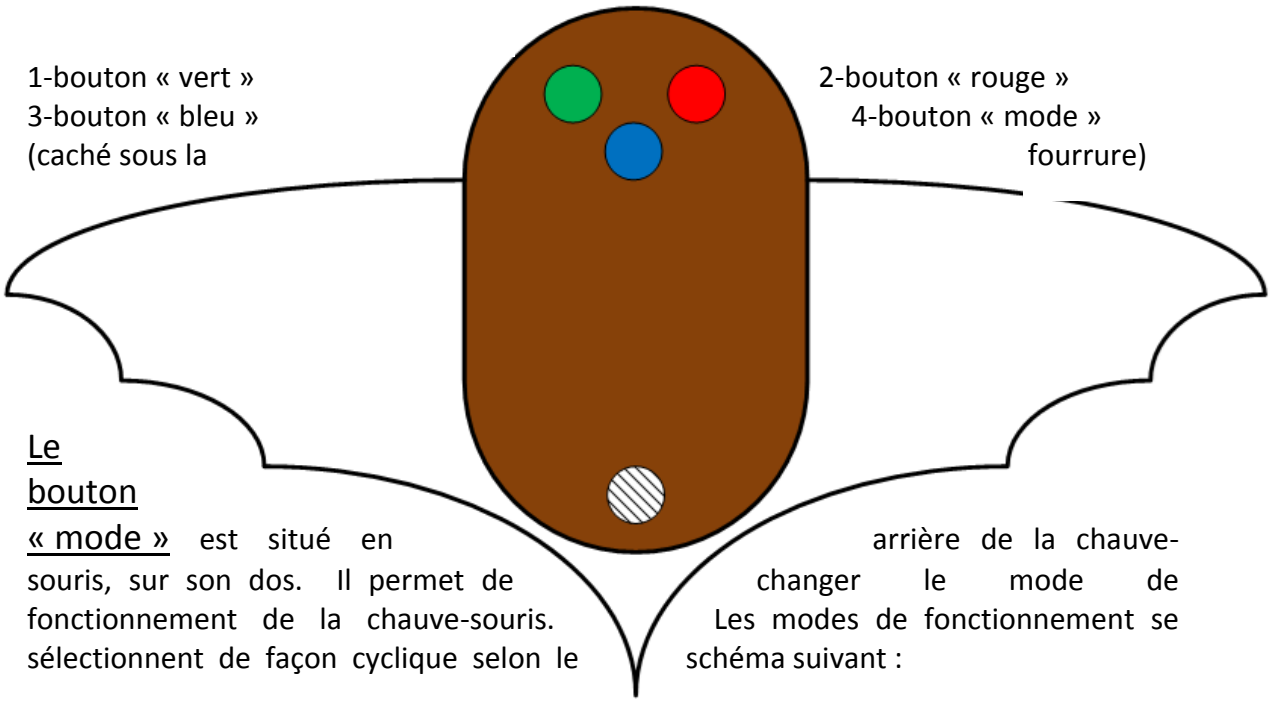
1. Pour faire démarrer la chauve-souris, il faut tout d'abord mettre l'interrupteur situé à l'arrière du robot sur la position active. Il est accessible en soulevant légèrement la coque. Il est assez gros et aisément identifiable. Les ailes ainsi que toutes les lumières vont se mettre en marche pendant le démarrage de celui-ci. Veuillez patienter.

2. Lorsque tout s'arrête, attendez encore environ 10 secondes afin de laisser le temps au programme de se charger.

3. Vous pouvez maintenant lancer le programme à l'aide du bouton situé à l'intérieur de la trappe dorsale. Il s'agit d'un tout petit bouton rouge situé sur la plaquette de montage des circuits. Une fois les yeux allumés par l'action de ce bouton, le robot est démarré en mode « Arrêté », expliqué à la page suivante.

Les boutons :

Le robot chauve-souris possède 4 boutons extérieurs:



<u>Le bouton « mode »</u> est situé en arrière de la chauve-souris, sur son dos. Il permet de changer le mode de fonctionnement de la chauve-souris. Les modes de fonctionnement se sélectionnent de façon cyclique selon le schéma suivant :

Mode Arrêté → Mode Évitement → Mode Chasse → Mode Questions → Mode Simon → Mode Danse → Mode aléatoire

Les boutons « bleu, rouge et vert » sont situés sur le front de la chauve-souris dans une disposition triangulaire et occupent diverses fonctions dépendamment du mode en cours.

Les modes :

Le mode « Arrêté » est le mode de départ et ne fait absolument rien. C'est en quelque sorte un mode « repos » où seulement les yeux sont allumés.

Le mode « Évitement » est le second mode et active le déplacement de la chauve-souris. Le déplacement se fait en ligne droite, jusqu'à ce qu'un obstacle soit détecté. À la détection d'un obstacle, celui-ci recule en faisant un arc afin de s'orienter dans une nouvelle direction, et ensuite repartir en ligne droite jusqu'au prochain obstacle détecté par les sonars dans les oreilles. En tout temps, la chanson « Slow ride » de *Foghat* joue, un bon vieux rock que l'on peut retrouver dans le jeu vidéo *Guitar Hero 3*.

Le mode « Chasse » est le troisième mode et correspond à l'inverse du précédent. La chauve-souris se déplace jusqu'à ce qu'elle détecte un obstacle. Elle se déplacera alors vers celui-ci et le suivra jusqu'à l'atteinte d'une distance minimale afin de ne pas foncer dessus. Si l'obstacle est passager (comme une personne qui passait devant par exemple), elle poursuivra son chemin. Si toutefois l'obstacle est fixe, elle attendra quelques instants, puis reculera en décrivant un arc pour ensuite partir dans une nouvelle direction. En tout temps, la chanson thème de l'émission « Batman » des années 70 joue.

Le mode « Questions » est le quatrième mode. La chauve-souris pose une question, puis invite l'enfant à y répondre par « Vrai » ou « Faux ». Durant l'attente de la réponse, la chanson thème de l'émission télévisée américaine « Jeopardy » joue, et les boutons vert (vrai) et rouge (faux) clignotent alternativement. Une fois la réponse donnée, le robot félicitera l'utilisateur par une commande vocale, un effet lumineux et un battement des ailes. Ensuite, une nouvelle question est posée. Si l'utilisateur ne répond pas avant la fin du thème musical, la question est répétée.

Le mode « Simon » est le cinquième mode et est une reprise du populaire jeu « Simon ». Il s'agit d'un jeu éducatif faisant appel à la mémoire. Le robot indique une séquence de couleurs ayant chacune leur son correspondant. Vous devez répéter celle-ci par les boutons bleu, vert et rouge.

Le mode « Danse » est le sixième mode. La chauve-souris exécute des mouvements aléatoires durant une chanson, afin d'imiter une danse. C'est purement un mode divertissant.

Le mode « Autonome » est le septième et dernier mode. La chauve-souris exécute les modes « Évitement », « Chasse » et « Questions » à tour de rôle.

Annexe 3 : Code Source du robot

```
/* Gotham Project: Kiwi la chauve-souris robotique
 * gothamproject.cpp
 *
 *  Version: 1-Dec-2008
 *      Author: Pascal Guay
 */

#include <libarmus.h> //Librairie armus
#include <fstream> //Librairie utilisée pour écrire dans un fichier
using namespace std;

//Liste des sorties des lumières
#define LRED 9 //DEL du bouton rouge
#define LGREEN 10 //DEL du bouton vert
#define LBLUE 11 //DEL du bouton bleu
#define LEYES 12 //DELS des yeux

//Liste des entrées des boutons
#define BGREEN 9 //Bouton vert
#define BRED 10 //Bouton rouge
#define BBLUE 11 //Bouton bleu
#define BMODE 12 //Bouton mode


#define FLY 16 //Sortie pour contrôler le battement des ailes
#define NQUEST 9 //Nombre de questions totales que le robot peut poser
#define SIMONTURN 999 //Nombre de séquences totales possibles  dans le jeu simon

/*Voici la classe robots, cette classe contient les variables du robot
 Une classe s'utilise en déclarant une objet comme étant une classe
 par exemple Robots kiwi; et la construction de la classe permet de bien organiser les
variables
 et de les utiliser de cette manière:  kiwi.follow = true par exemple permet de mettre le
robot en mode suivi.
*/

class Robots
{
public:
      int sonar1; //Distance du sonar gauche
      int sonar2; //Distance du sonar droit
      int speedleft; //Vitesse du moteur gauche
      int speedright; //Vitesse du moteur droit
      int mindist; // Distance minimum utilisé pour éviter les obstacles
      int blinkmode; //Variable de changement de mode de clignotement des lumières
      int mode; //Variable de changement de mode
      bool avoid; //Variable du mode évitement des obstacles
      bool follow; //Variable du mode suivie
      bool askquestion; //Variable du mode question
      bool simon; //Variable du mode jeu Simon
      bool dance; //Variable du mode danser
      bool autonomous; //Variable du mode autonomie
      int getSonar(int); //Cette fonction sert à obtenir la distance du sonar (1 ou 2)
selon le paramètre
```

```
      void setSpeed(int,int); //Cette fonction sert à modifier la vitesse du robot, le
moteur gauche en premier et le droit en deuxième paramètres
      void setPin(int, bool); //Cette fonction sert à modifier l'état des pins sur la
carte I/O
      int getSpeedleft(); //Cette fonction sert à obtenir la vitesse du moteur gauche
      int getSpeedright(); //Cette fonction sert à obtenir la vitesse du moteur droit
      bool getButtonState(int); //Fonction afin d'obtenir l'état des boutons, le numéro de
la sortie étant appelé en paramètre.
      void reset();
};
Robots kiwi; //Ici on déclare l'objet kiwi qui est membre de la classe Robots
void avoid(); //Fonction d'évitement
void follow(); //Fonction de suivie
void askquestion(); //Fonction pour poser des questions
void simon(); //Fonction du jeu simon
void dance(); //Fonction de danse
void autonomous(); //Fonction autonome
void writeanswer(int,bool); //Fonction pour écrire les statistiques des réponses des
enfants dans un fichier.
void writesimon(int); //Fonction pour écrire le score des jours du jeu simon dans un
fichier
void blink(); //Fonction des différents mode de clignottement des LEDS
void setMode(int); //Fonction servant à changer le mode du robot

//Voici les différents thread qui pourront s'exécuter en parallèle
THREAD thread_avoid;
THREAD thread_askquestion;
THREAD thread_follow;
THREAD thread_simon;
THREAD thread_blink;
THREAD thread_dance;
THREAD thread_autonomous;

int main()
{
      ERROR_SetMode(LCD_ONLY); //Permet de visualiser les erreurs sur le LCD

      thread_blink  = THREAD_CreateSimple(blink); //Création du thread pour faire
clignoter les lumières
      AUDIO_SetVolume(100); //Mise du volume au maximum
      int mode; //Variable permettant de changer de mode
      int timemusic = 0; //Variable permettant de stocker le lapse de temps depuis le
début de la musique
      int musicduration = 0; //Variable permettant de stocker le temps de la musique
      kiwi.setPin(LEYES, 1); //Les yeux de la chauve-souris se mettent en fonction
      while(1){ //Boucle de la fonction main
            if(kiwi.getButtonState(BMODE)){ //Attente d'appuie sur le bouton mode

            mode++;
            if(mode == 1){ //Si c'est le mode évitement obstacle
                  timemusic = 0;
            musicduration = 24000; //Le fichier explication.mp3 dure 24 secondes
            }
            if(mode == 2){ //Si c'est le mode suivi
                  timemusic = 0;
            musicduration = 55000; //Musique batman.mp3 dure 55 secondes
            }
            if(mode == 6){ //Si c'est le mode danse
                  timemusic = 0;
```

```
            musicduration = 257000; //Chanson canttouchthis dure 257 secondes
            }
            setMode(mode); //On active le mode
            if(mode > 6){ //Si le mode dépasse le nombre de mode
                  mode = 0;
            }
            THREAD_MSleep(1000);
            }
//Code afin de répéter la musique
            if(mode == 1){ //Si c'est le mode suivi
                  timemusic += 100; //On additionne une variable temps
                  if(timemusic > musicduration){ //Si la variable temps dépasse la durée
de la musique
                              AUDIO_PlayMP3File("slowride.mp3"); //On fait rejouer la
musique en boucle
                              timemusic = 0;
                              musicduration = 238000;
                  }
            }
            if(mode == 2){
                  timemusic += 100;
                  if(timemusic > musicduration){
                              AUDIO_PlayMP3File("batman.mp3");
                              timemusic = 0;
                              musicduration = 55000;
                  }
            }

            if(mode == 6){
                  timemusic += 100;
                  if(timemusic > musicduration){
                              AUDIO_PlayMP3File("canttouchthis.mp3");
                              timemusic = 0;
                              musicduration = 257000;
                  }
            }
//Fin du code de répétition de la musique
            THREAD_MSleep(100); //Chaque boucle doit avoir une fonction permettant de la
faire dormir afin de ne pas prendre 100% du CPU.
      }
      FPGA_StopAll(); //Arrête les circuits intégrés de la carte armus.
      return 0;
}

/*
 * Ceci est la fonction permettant de changer de mode, en appuyant sur le bouton mode
 * cette fonction est appelée avec en paramètre le numéro du mode
 */

void setMode(int mode){
      switch (mode){
      case 1: //Mode évitement
      kiwi.reset(); //Permet de réinitialiser les variables du robot
      thread_avoid  = THREAD_CreateSimple(avoid); //On crée le thread d'évitement
d'obstacle
      AUDIO_PlayMP3FileAndWait("modeevitement.mp3"); //Parole permettant de savoir dans
quel mode le robot vient de se mettre
      AUDIO_PlayMP3File("explication.mp3"); //Le robot explique le principe des sonars
d'une chauve-souris
```

```
      kiwi.blinkmode = 3; //Mode de clinotement des DELS aléatoire
      kiwi.setPin(FLY,1); //On fait battre des ailes le robot
      kiwi.avoid = true; //On met la variable avoid à true
      break; //On quitte le switch
      //Les mêmes types de comementaires s'appliquent pour le reste des modes
      case 2: //Mode suivi
            THREAD_Destroy(&thread_avoid); //On détruit la fonction d'évitement afin
qu'elle n'agisse plus sur le robot
            kiwi.reset();
            thread_follow  = THREAD_CreateSimple(follow);
            AUDIO_PlayMP3FileAndWait("modesuivi.mp3");
            AUDIO_PlayMP3File("batman.mp3");
            kiwi.blinkmode = 0;
            kiwi.setPin(FLY,1);
            kiwi.follow = true;
      break;
      case 3: //Mode questions
            THREAD_Destroy(&thread_follow);
            kiwi.reset();
            thread_askquestion  = THREAD_CreateSimple(askquestion);
            AUDIO_PlayMP3FileAndWait("modequestion.mp3");
            AUDIO_PlayMP3File("quiz.mp3");
            kiwi.askquestion = true;
      break;
      case 4: //Mode jeu Simon
            THREAD_Destroy(&thread_askquestion);
            kiwi.reset();
            thread_simon  = THREAD_CreateSimple(simon);
            AUDIO_PlayMP3FileAndWait("modesimon.mp3");
            kiwi.simon = true;
            break;
      case 5: //Mode danser
            THREAD_Destroy(&thread_simon);
            kiwi.reset();
            thread_dance  = THREAD_CreateSimple(dance);
            AUDIO_PlayMP3FileAndWait("modealeatoire.mp3");
            kiwi.blinkmode = 4;
            kiwi.setPin(FLY,1);
            kiwi.dance = true;
            AUDIO_PlayMP3File("canttouchthis.mp3");
            break;
      case 6: //Mode autonomie
            THREAD_Destroy(&thread_dance);
            kiwi.reset();
            AUDIO_PlayMP3FileAndWait("modeautonome.mp3");
            thread_autonomous  = THREAD_CreateSimple(autonomous);
            kiwi.autonomous = true;
            break;
      case 7: //Mode arrêt
            THREAD_Destroy(&thread_autonomous); //On détruit les autres modes
            THREAD_Destroy(&thread_avoid);
            THREAD_Destroy(&thread_follow);
            THREAD_Destroy(&thread_askquestion);
            THREAD_Destroy(&thread_dance);
            kiwi.reset();
            AUDIO_PlayMP3FileAndWait("modearret.mp3");
            break;
      }
}
```

```
void avoid() //Fonction d'évitement d'objets
{
	int time = 0; //Temps mis par le robot dans sa phase de contournement d'objets
	kiwi.mindist = 35; //Distance minimale d'évitement
while(1){
	if(!kiwi.avoid){ //Si la fonction n'est pas enclenchée alors on fait la fait dormir.
		THREAD_MSleep(1000);
	}else{
		THREAD_MSleep(10);
		if(kiwi.getSonar(1) < kiwi.mindist && kiwi.getSonar(1) != 0){ //Si le sonar
gauche est plus petit que la distance minimale
			while(time <= 6){
				time++;
				if(time < 4){
				kiwi.setSpeed(90,90); //On recule pendant un certain temps
				}
				if(time >= 4){
				kiwi.setSpeed(0,100); //On tourne à droite
				}
				THREAD_MSleep(500);
			}
			time = 0;
		}else if(kiwi.getSonar(2) < kiwi.mindist && kiwi.getSonar(2) != 0){ //Si le
sonar droit est plus petit que la distance minimale
			while(time <= 6){
				time++;
				if(time < 4){
				kiwi.setSpeed(90,90); //On recule pendant un certain temps
				}
				if(time >= 4){
				kiwi.setSpeed(100,0); //On tourne à droite
				}
				THREAD_MSleep(500);
			}
			time = 0;
		}else{
			kiwi.setSpeed(-80,-80); // On avance tout droit si la distance est assez
grande
		}
	}
}
}

void follow() //Fonction de suivi d'objets
{
	int timestop = 0; //Permet d'évaluer le temps d'arrêt du robot pour éviter de suivre
un objet qui est fixe.
while(1){
	if(!kiwi.follow){
		THREAD_MSleep(1000);
	}else{
		THREAD_MSleep(10);
			while(kiwi.getSonar(1) < kiwi.getSonar(2) &&
SYSTEM_Abs(kiwi.getSonar(1)-kiwi.getSonar(2)) > 25){ //Comparaisons de la distance entre
les deux sonars
				kiwi.setSpeed(0,-75); //Tourne à gauche pour se positionner sur
l'objet le plus proche si la distance du sonar gauche est la plus petite
```

```
                        THREAD_MSleep(5);
                  }
                        while(kiwi.getSonar(1) > kiwi.getSonar(2) &&
SYSTEM_Abs(kiwi.getSonar(1)-kiwi.getSonar(2)) > 25){ //Comparaisons de la distance entre
les deux sonars
                              kiwi.setSpeed(-75,0); //Tourne à droite pour se positionner
sur l'objet le plus proche si la distance du sonar droit est la plus petite entre les
deux
                              THREAD_MSleep(5);
                        }
            if(kiwi.getSonar(1) > 40 && kiwi.getSonar(2) > 40){ //Si la distance des deux
sonars est plus grande que 40 alors
            kiwi.setSpeed(-80, -80); //On fait avancer tout droit
            }else{
                  timestop += 100; //On additionne depuis combien de temps le robot est en
arrêt sur la cible poursuivi
                  if(timestop > 1500){ //Après un certains temps on fait reculer et
tourner le robot
                        timestop = 0;
                        while(timestop <= 6){
                              if(timestop < 4){
                              kiwi.setSpeed(100,100); //Le robot recule pendant un certain
temps
                              }
                              if(timestop >= 4){
                              kiwi.setSpeed(100,0); //Le robot se tourne
                              }
                        THREAD_MSleep(500);
                        timestop++;
                  }
                        timestop = 0;
                  }
            kiwi.setSpeed(0, 0); //Sinon on arrête le robot

            }
      }
}
}

void askquestion() //Fonction questions
{
      int numquestion = 0; //Variable pour tenir en mémoire à quel questions on est rendu
      int threequestion = 0; //Variable servant pour poser 3 questions dans le mode
autonome
      int TableQuestion[NQUEST] = {0}; //Table aléatoire des numéros de questions

      bool answer; //Variable stockant la réponse à la question
      bool goodanswer; //Variable de l'enfant permettant de savoir si sa réponse est
exacte

      while(1)
      {
            if(!kiwi.askquestion)
            {
                  int numquestion = 0;
                  int threequestion = 0;
                  int TableQuestion[NQUEST] = {0};
                  THREAD_MSleep(1000);
            }else{
```

```
                  if(numquestion == 0){  //Si nous sommes à la première question le robot
se présente et explique à l'enfant le principe de ce mode
                        AUDIO_PlayMP3FileAndWait("bonjour.mp3");
                        AUDIO_PlayMP3FileAndWait("appellekiwi.mp3");
                        if(kiwi.autonomous){
                        AUDIO_PlayMP3FileAndWait("3questions.mp3");
                        }else{
                        AUDIO_PlayMP3FileAndWait("9questions.mp3");
                  }
                        AUDIO_PlayMP3FileAndWait("boutonsfauxvrai.mp3");
                        AUDIO_PlayMP3FileAndWait("boutonrepete.mp3");
                  }
                  /*if(numquestion == 0){
                        int numquestion = 0;
                        int threequestion = 0;
                        int TableQuestion[NQUEST] = {0};
                  }*/
                  TableQuestion[numquestion] = SYSTEM_Random(NQUEST); //On génère le
nombre aléatoire
                  for(int i = 0; i <= numquestion; i++) //Boucle servant à vérifier si la
question à déjà été posé afin de générer un numéro aléatoire pour une question qui n'a
pas été posée
                  {
                        if(numquestion != 0){
                        if(TableQuestion[numquestion] == TableQuestion[i - 1]) //Est-ce
que les questions sont les mêmes ?
                        {
                              TableQuestion[numquestion] = SYSTEM_Random(NQUEST); //On
regénère un numéro de question aléatoire
                              i = 0; //On recommence la boucle
                        }
                  }
                  }
                  repeat: //Saut permettant de répéter la question au cas où l'enfant sur
le bouton bleu

            switch (TableQuestion[numquestion]){ //Le robot pose la question
correspondante à l'enfant
            case 0:
                  AUDIO_PlayMP3FileAndWait("question1.mp3");
                  answer = true; //La réponse à cette question est vraie
                  break;
            case 1:
                  AUDIO_PlayMP3FileAndWait("question2.mp3");
                  answer = true;
                  break;
            case 2:
                  AUDIO_PlayMP3FileAndWait("question3.mp3");
                  answer = true;
                  break;
            case 3:
                  AUDIO_PlayMP3FileAndWait("question4.mp3");
                  answer = true;
                  break;
            case 4:
                  AUDIO_PlayMP3FileAndWait("question5.mp3");
                  answer = false; //La réponse à cette question est fausse
                  break;
            case 5:
```

```
                  AUDIO_PlayMP3FileAndWait("question6.mp3");
                  answer = false;
                  break;
            case 6:
                  AUDIO_PlayMP3FileAndWait("question7.mp3");
                  answer = false;
                  break;
            case 7:
                  AUDIO_PlayMP3FileAndWait("question8.mp3");
                  answer = false;
                  break;
            case 8:
                  AUDIO_PlayMP3FileAndWait("question9.mp3");
                  answer = true;
                  break;
            }
            AUDIO_PlayMP3FileAndWait("vraioufaux.mp3"); // Le robot dit "vrai ou faux"
            AUDIO_PlayMP3File("quiz.mp3"); //Une musique joue pour attendre que l'enfant
appui sur le bouton
            kiwi.blinkmode = 1; //Mode clignotement des lumières vrai faux
            int randomvar = SYSTEM_Random(3); //Nombre aléatoire servant à exprimer
différentes réactions au robot
            int timewait = 0;
            while(kiwi.askquestion){
                  if(kiwi.getButtonState(BGREEN)){ //Vérification du bouton vrai
                        AUDIO_StopPlayback(); //On arrête la musique d'attente
                        kiwi.blinkmode = 0;
                        if(answer == true){
                              kiwi.setPin(FLY, 1); //Le robot bat des ailes pour féliciter
l'enfant
                              kiwi.setPin(LRED, 0);
                              kiwi.setPin(LGREEN, 1);
                              goodanswer = true; //L'enfant a choisi la bonne réponse
                              AUDIO_PlayMP3FileAndWait("bonnereponse.mp3");;
                              switch (randomvar){ //Le robot félicite l'enfant selon
plusieurs possiblités
                              case 0:
                                    AUDIO_PlayMP3FileAndWait("bravo.mp3");
                                    AUDIO_PlayMP3FileAndWait("rire1.mp3");
                                    break;
                              case 1:
                                    AUDIO_PlayMP3FileAndWait("felicitation.mp3");
                                    AUDIO_PlayMP3FileAndWait("rire3.mp3");
                                    break;
                              case 2:
                                    AUDIO_PlayMP3FileAndWait("excellent.mp3");
                                    AUDIO_PlayMP3FileAndWait("rire4.mp3");
                                    kiwi.setPin(LGREEN, 0);
                                    break;
                              }
                              kiwi.setPin(LGREEN, 0);
                              kiwi.setPin(FLY, 0);
                              break;
                        }else{ //La mauvaise réponse
                              kiwi.setPin(LRED, 0);
                              kiwi.setPin(LGREEN, 1);
                              goodanswer = false; //L'enfant a choisi la mauvaise réponse
                              AUDIO_PlayMP3FileAndWait("mauvaisereponse.mp3"); //Le robot
dit "mauvaise réponse".
```

```
                        kiwi.setPin(LGREEN, 0);
                        break;
                  }
            }
            if(kiwi.getButtonState(BRED)){ //  vérification du bouton faux
                  AUDIO_StopPlayback();
                  kiwi.blinkmode = 0;
                  if(answer == false){ //Vérification si l'enfant à la bonne réponse
                        kiwi.setPin(FLY, 1); //On bat des ailes et une voix
aléatoire félicite l'enfant
                        kiwi.setPin(LGREEN, 0);
                        kiwi.setPin(LRED, 1);
                        goodanswer = true;
                        AUDIO_PlayMP3FileAndWait("bonnereponse.mp3");
                        switch (randomvar){
                        case 0:
                              AUDIO_PlayMP3FileAndWait("bravo.mp3");
                              AUDIO_PlayMP3FileAndWait("rire7.mp3");
                              break;
                        case 1:
                              AUDIO_PlayMP3FileAndWait("felicitation.mp3");
                              AUDIO_PlayMP3FileAndWait("rire8.mp3");
                              break;
                        case 2:
                              AUDIO_PlayMP3FileAndWait("excellent.mp3");
                              AUDIO_PlayMP3FileAndWait("rire10.mp3");
                              break;
                        }
                        kiwi.setPin(LRED, 0);
                        kiwi.setPin(FLY, 0);
                  }else{ //Mauvaise réponse
                        kiwi.setPin(LGREEN, 0);
                        kiwi.setPin(LRED, 1);
                        goodanswer = false;
                        AUDIO_PlayMP3FileAndWait("mauvaisereponse.mp3");
                        kiwi.setPin(LRED, 0);
                  }
                  break;
      }
            if(kiwi.getButtonState(BBLUE)){ //Si l'enfant appuie sur le bouton bleu
            goto repeat; //On répète la question
            break;
            }
            timewait += 1; //Si la musique d'attente se termine on répète la
question
            if(timewait > 370){
                  goto repeat;
                  break;
            }
            THREAD_MSleep(100); //On oublie pas de mettre un sleep dans les boucles
afin de ne pas utiliser 100% du CPU
            }
      writeanswer(TableQuestion[numquestion], goodanswer); //On écrit la réponse de
l'enfant dans un fichier statistique

      switch (TableQuestion[numquestion]){ // On dit plus de détail sur la question
posée à l'enfant
      case 0:
            AUDIO_PlayMP3FileAndWait("reponse1.mp3");
```

```
                  break;
            case 1:
                  AUDIO_PlayMP3FileAndWait("reponse2.mp3");
                  break;
            case 2:
                  AUDIO_PlayMP3FileAndWait("reponse3.mp3");
                  break;
            case 3:
                  AUDIO_PlayMP3FileAndWait("reponse4.mp3");
                  break;
            case 4:
                  AUDIO_PlayMP3FileAndWait("reponse5.mp3");
                  break;
            case 5:
                  AUDIO_PlayMP3FileAndWait("reponse6.mp3");
                  break;
            case 6:
                  AUDIO_PlayMP3FileAndWait("reponse7.mp3");
                  break;
            case 7:
                  AUDIO_PlayMP3FileAndWait("reponse8.mp3");
                  break;
            case 8:
                  AUDIO_PlayMP3FileAndWait("reponse9.mp3");
                  break;
      }
            numquestion++; //On incrémente à la prochaine question

            if(kiwi.autonomous){ //Si le robot est en mode autonome on pose seulement
trois questions
                  threequestion++;
            }
            if(numquestion > NQUEST-1){ //Si le robot a posé les 9 questions on revient au
début de la fonction
                  numquestion = 0;
                  threequestion = 0;
            }

            if(threequestion > 2){ //Si le robot a poser trois questions en mode autonome
alors on met le mode question à OFF
                  threequestion = 0;
                  kiwi.askquestion  = false;
            }
}
}
}

void simon() //Fonction du jeu de répétition de séquences Simon
{
      kiwi.blinkmode = 0;
      bool lose = false; //Variable stockant si l'enfant à perdu
      int memturn = 0; //Variable servant à stocker le nombre de séquences atteint par
l'enfant
      int turn = 0; //Variable servant à stocker la position dans la séquence

      int tableturn[SIMONTURN] = {0}; //Initialisation de la table de stockage des
séquences
      while(1)
      {
```

```
            if(!kiwi.simon){
                  THREAD_MSleep(1000);
            }else{
                  if(memturn == 0){ //Si nous sommes à la première séquence, le robot
explique à l'enfant comment jouer
                        AUDIO_PlayMP3FileAndWait("joueavecmoi.mp3");
                        AUDIO_PlayMP3FileAndWait("explicationsimon.mp3");
                  }
                  THREAD_MSleep(750);
                  tableturn[memturn] = SYSTEM_Random(3); //Détermination aléatoire d'une
séquence
                  //Le robot fait jouer la séquence avec un son, un bouton et une lumière
correspondante
      for(int i = 0; i <= memturn; i++){
            switch (tableturn[i]){
            case 0: //Bouton rouge
                  kiwi.setPin(LRED, 1); //Allume la DEL du bouton rouge
                  kiwi.setPin(LGREEN, 0);
                  kiwi.setPin(LBLUE, 0);
                  AUDIO_PlayMP3File("bouton1.mp3"); //Fait jouer le bruit correspondant
                  THREAD_MSleep(750);
                  kiwi.setPin(LRED, 0);
                  THREAD_MSleep(100);
                  break;
            case 1: //Bouton vert
                  kiwi.setPin(LRED, 0);
                  kiwi.setPin(LGREEN, 1);
                  kiwi.setPin(LBLUE, 0);
                  AUDIO_PlayMP3File("bouton2.mp3");
                  THREAD_MSleep(750);
                  kiwi.setPin(LGREEN, 0);
                  THREAD_MSleep(100);
                  break;
            case 2: //Bouton bleu
                  kiwi.setPin(LRED, 0);
                  kiwi.setPin(LGREEN, 0);
                  kiwi.setPin(LBLUE, 1);
                  AUDIO_PlayMP3File("bouton3.mp3");
                  THREAD_MSleep(750);
                  kiwi.setPin(LBLUE, 0);
                  THREAD_MSleep(100);
                  break;
            }
      }
      //Attente d'appuie sur un bouton
      while(turn <= memturn && kiwi.simon){ //Boucle vérifiant si la séquence qui doit
être répétée est bonne
if(kiwi.getButtonState(BRED)){ //Le bouton rouge est appuyé
      if(tableturn[turn] == 0){ //On vérifie si c'est bien la bonne séquence
            turn++; //On additionne la position où l'enfant est dans la séquence
            kiwi.setPin(LRED, 1); //On fait allumer le bouton rouge
            AUDIO_PlayMP3File("bouton1.mp3"); //On fait jouer le son correspondant
            THREAD_MSleep(250);
            kiwi.setPin(LRED, 0);
      }else{ //Si ce n'est pas le bon bouton qui a été appuyé
lose = true; //L'enfant perd
break;
      }
}
```

```
if(kiwi.getButtonState(BGREEN)){ //Le bouton vert est appuyé
      if(tableturn[turn] == 1){
            turn++;
            kiwi.setPin(LGREEN, 1);
            AUDIO_PlayMP3File("bouton2.mp3");
            THREAD_MSleep(250);
            kiwi.setPin(LGREEN, 0);
      }else{
lose = true;
break;
      }
}
if(kiwi.getButtonState(BBLUE)){ //Le bouton bleu est appuyé
      if(tableturn[turn] == 2){
            turn++;
            kiwi.setPin(LBLUE, 1);
            AUDIO_PlayMP3File("bouton3.mp3");
            THREAD_MSleep(250);
            kiwi.setPin(LBLUE, 0);
      }else{
lose = true;
break;
      }
}
THREAD_MSleep(50);
      }
                  if(lose) //Si la personne n'a pas fait la bonne séquence
                  {
                        AUDIO_PlayMP3FileAndWait("pasbonnesequence.mp3"); //Le robot parle
pour dire que ce n'était pas la bonne séquence

                        if(memturn>=9){ //Si le score est plus haut que 10 séquences
                              AUDIO_PlayMP3FileAndWait("felicitationbonscore.mp3"); //On
dit à l'enfant que son score est très bon
                        }else{
                              AUDIO_PlayMP3FileAndWait("bravoamelioration.mp3"); //Sinon
on dit à l'enfant "Bravo, mais il y a place à amélioration"
                        }
                  writesimon(memturn-1); //On écrit le score que obtenu par l'enfant dans
un fichier
                  kiwi.setPin(FLY, 1); //On fait battre des ailes et clignoter les
lumières du robot
                  kiwi.blinkmode = 2;
                  THREAD_MSleep(1000);
                  kiwi.blinkmode = 4;
                  THREAD_MSleep(1000);
                  kiwi.blinkmode = 0;
                  THREAD_MSleep(1000);
                  kiwi.setPin(FLY, 0);
                  turn = 0; //On réinitialise tout pour que le robot pour continuer une
nouvelle partie
                  memturn = -1;
                  lose = false;
                  THREAD_MSleep(1000);
                  }
                  memturn++;
                  turn = 0;

            }
```

```
	}
}
void dance() //Fonction danser
{
	kiwi.mindist = 20; //Distance minimale d'évitement
while(1){
	if(!kiwi.dance){
		THREAD_MSleep(1000);
	}else{
		THREAD_MSleep(50);
		if(kiwi.getSonar(1) < kiwi.mindist && kiwi.getSonar(1) != 0){ //Si le sonar
gauche est plus petit que la distance minimale
			kiwi.setSpeed(SYSTEM_Random(90),SYSTEM_Random(90)); // On tourne à
droite aléatoirement pendant un court lapse de temps
			THREAD_MSleep(250);
		}else if(kiwi.getSonar(2) < kiwi.mindist && kiwi.getSonar(2) != 0){ //Si le
sonar droit est plus petit que la distance minimale
			kiwi.setSpeed(SYSTEM_Random(90),SYSTEM_Random(90)); // On tourne à
gauche aléatoirement pendant un court lapse de temps
			THREAD_MSleep(250);
		}else{
			kiwi.setSpeed(-SYSTEM_Random(90), -SYSTEM_Random(90)); // On fait
avancer le robot par petit pas aléatoires
		}
	}
}
}

void autonomous(){ //Fonction d'autonomie
	while(1){
		if(!kiwi.autonomous){
			THREAD_MSleep(1000);
		}else{
			kiwi.reset(); //Le robot passe en mode évitement d'objet pendant 10
secondes
			kiwi.avoid = true;
			kiwi.setPin(FLY,1);
			thread_avoid  = THREAD_CreateSimple(avoid);
			kiwi.blinkmode = 3;
			AUDIO_PlayMP3File("slowride.mp3");
				THREAD_MSleep(10000);
				THREAD_Destroy(&thread_avoid);
				kiwi.reset();
				AUDIO_PlayMP3File("batman.mp3");
				kiwi.setPin(FLY,1);
				kiwi.blinkmode = 3;
				kiwi.follow= true; //Le robot passe en mode suivie pendant
10 secondes
				thread_follow  = THREAD_CreateSimple(follow);
				THREAD_MSleep(10000);
				THREAD_Destroy(&thread_follow);
				kiwi.reset();
				kiwi.askquestion = true; //Le robot pose trois questions
				thread_askquestion  = THREAD_CreateSimple(askquestion);
				while (kiwi.askquestion){
					THREAD_MSleep(1000);
				}
				THREAD_Destroy(&thread_askquestion);
				kiwi.reset();
```

```
                                    kiwi.setPin(FLY,1);
                                    kiwi.blinkmode = 4;
                                    AUDIO_PlayMP3File("canttouchthis.mp3");
                                    kiwi.dance = true; //Le robot danse
                                    thread_dance  = THREAD_CreateSimple(dance);
                                    THREAD_MSleep(10000);
                                    THREAD_Destroy(&thread_dance);
                  }
            }

}
void blink(){ //Fonction de clingotement des lumières
while(1){
      switch (kiwi.blinkmode){
      case 0:
            THREAD_MSleep(250);
            break;
      case 1: //mode de clignotement vrai ou faux
            kiwi.setPin(LGREEN,0);
            kiwi.setPin(LRED,1);
            THREAD_MSleep(1000);
            kiwi.setPin(LRED,0);
            kiwi.setPin(LGREEN,1);
            THREAD_MSleep(1000);
            break;
      case 2: //Lorsqu'on gagne on fait clignoter les dels de cette manière
            kiwi.setPin(LRED,1);
            THREAD_MSleep(100);
            kiwi.setPin(LRED,0);
            kiwi.setPin(LBLUE,1);
            THREAD_MSleep(100);
            kiwi.setPin(LBLUE,0);
            kiwi.setPin(LGREEN,1);
            THREAD_MSleep(100);
            kiwi.setPin(LGREEN,0);
            break;
      case 3: //Fonction aléatoire de clinotement des dels
            kiwi.setPin(LRED,SYSTEM_Random(2));
            kiwi.setPin(LBLUE,SYSTEM_Random(2));
            kiwi.setPin(LGREEN,SYSTEM_Random(2));
            kiwi.setPin(LEYES,SYSTEM_Random(2));
            THREAD_MSleep(500);
            break;

      case 4: //Fonction clingotement stroboscopique
            kiwi.setPin(LRED,1);
            kiwi.setPin(LBLUE,1);
            kiwi.setPin(LGREEN,1);
            kiwi.setPin(LEYES,1);
            THREAD_MSleep(500);
            kiwi.setPin(LRED,0);
            kiwi.setPin(LBLUE,0);
            kiwi.setPin(LGREEN,0);
            kiwi.setPin(LEYES,0);
            THREAD_MSleep(500);
            break;
      }
}
}
```

```
void Robots::reset(){ //Fonction servant à réinitialiser les variables du robot
	kiwi.setSpeed(0,0);
	kiwi.avoid = false;
	kiwi.follow = false;
	kiwi.askquestion  = false;
	kiwi.simon = false;
	kiwi.dance = false;
	kiwi.blinkmode = 0;
	kiwi.setSpeed(0,0);
	kiwi.setPin(LEYES,1);
	kiwi.setPin(FLY,0);
	kiwi.setPin(LRED, 0);
	kiwi.setPin(LBLUE, 0);
	kiwi.setPin(LGREEN, 0);
	AUDIO_StopPlayback();
}

bool Robots::getButtonState(int num) //Vérification de l'état d'un bouton
{
	if(DIGITALIO_Read(num)){
		return true; //Retourne vrai si le bouton est appuyé
	}else{
		return false; //Retourne faux si le bouton n'est pas appuyé
	}
}


void writeanswer(int question, bool success){ //Fonction servant à écrire les
statistiques dans le fichier
ofstream OutputFile("/media/usb0/reponses.txt", ios::app); //On ouvre le fichier en se
positionnant à la fin avec la commande ios::app
if(success){  //Si l'enfant a répondu la bonne réponse on écrit dans le fichier le # de
la question suivi du chiffre 1.
OutputFile << question++ << ":" << 1 << " ";
}else{
	//Si l'enfant n'a pas répondu la bonne réponse on écrit dans le fichier le # de la
question suivi du chiffre 0.
	OutputFile << question++ << ":" << 0 << " ";
}
OutputFile.close(); //On ferme le fichier
}
void writesimon(int score){ //Fonction servant à écrire les statistiques dans le fichier
ofstream OutputFile("/media/usb0/simon.txt", ios::app); //On ouvre le fichier en se
positionnant à la fin avec la commande ios::app
OutputFile  << score << ","; //On inscrit le score obtenu séparé par une virgule
OutputFile.close();
}
int Robots::getSpeedleft() //Fonction renvoyant la vitesse gauche du moteur
{
	return kiwi.speedleft;
}

int Robots::getSpeedright() //Fonction renvoyant la vitesse droit du moteur
{
	return kiwi.speedright;
}
```

```
void Robots::setPin(int pin, bool state){ //Fonction permettant de controller les sorties
	DIGITALIO_Write(pin, state);
}
void Robots::setSpeed(int speedleft, int speedright) //Fonction servant à donner la
vitesse aux moteurs
{
	MOTOR_SetSpeed(MOTOR_LEFT,  speedleft); //Vitesse moteur gauche
	kiwi.speedleft = speedleft;
	MOTOR_SetSpeed(MOTOR_RIGHT, speedright); //Vitesse moteur droit
	kiwi.speedright = speedright;
}

int Robots::getSonar(int sonarnumber) //Fonction servant à obtenir la ditance du moteur
{
	THREAD_MSleep(65); //Délai nécessaire pour ne pas que les sonars entrent en conflit
	return SONAR_Detect(sonarnumber); //On renvoie la distance captée par le sonar
}
```